Agentic coding: the next frontier

From autocomplete to autonomous

AGENDA

What we'll cover

01

Before Ignition

AI coding tools existed - but they weren't going anywhere on their own

02

Leaving the atmosphere

Not all AI coding is built for orbit

03

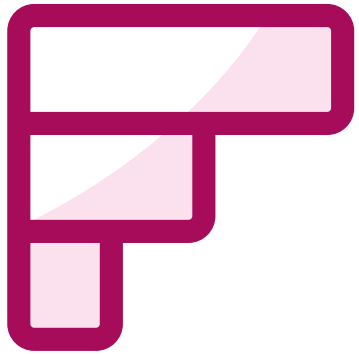
Reaching Orbit

How agentic coding works

04

Touchdown

Equipping the crew for the terrain ahead



How are you using AI for development?

01 Before Ignition

AI coding tools existed - but they weren’t going anywhere on their own

Not so long ago...

AI brought speed to individual tasks - but couldn't think across your codebase

✓

- Answer questions
- Inline suggestions
- Single-file edits
- Great at C#, Python, JavaScript..

Useful day-to-day

✗

- Multi-file reasoning
- Long-running tasks
- Codebase awareness
- Struggled with AL, niche ERPs

Fundamentally limited

A smart search engine. Useful – but you were still doing all the work.

The models caught up

Five shifts that turned assistants into agents.

CONTEXT	RETRIEVAL	REASONING	HORIZON	TOOL USE
1M	**76%**	**4**	**30h**	**100s**
tokens in context	at full window	effort levels	per task	tools per task


02
Leaving the atmosphere
Not all AI coding is built for orbit

What is vibe coding?

- Embrace the exponentials
- Forget the code exists
- Extensive use of AI to generate your code

Vibe coding raises the bar for what everyone can do in software

What you can ship with vibe coding

Describe it. Deploy it. Before the kick-off meeting finishes.

Landing pages

HOURS

A campaign page for a product launch, ready before the email goes out.

Workflow automations

AN AFTERNOON

A bot that routes customer tickets to the right team, no ops backlog.

Internal tools

A DAY

An ops dashboard for the finance team - live, queryable, no ticket queue.

MVPs & prototypes

DAYS

A clickable product to put in front of real users before you invest in a build.

Used by founders, PMs, designers – not just engineers

Lots of examples of this going wrong...

Vibe coding is more capable than you think - and that's the risk

The wall isn't what it can build. It's what it leaves behind.

What it can do

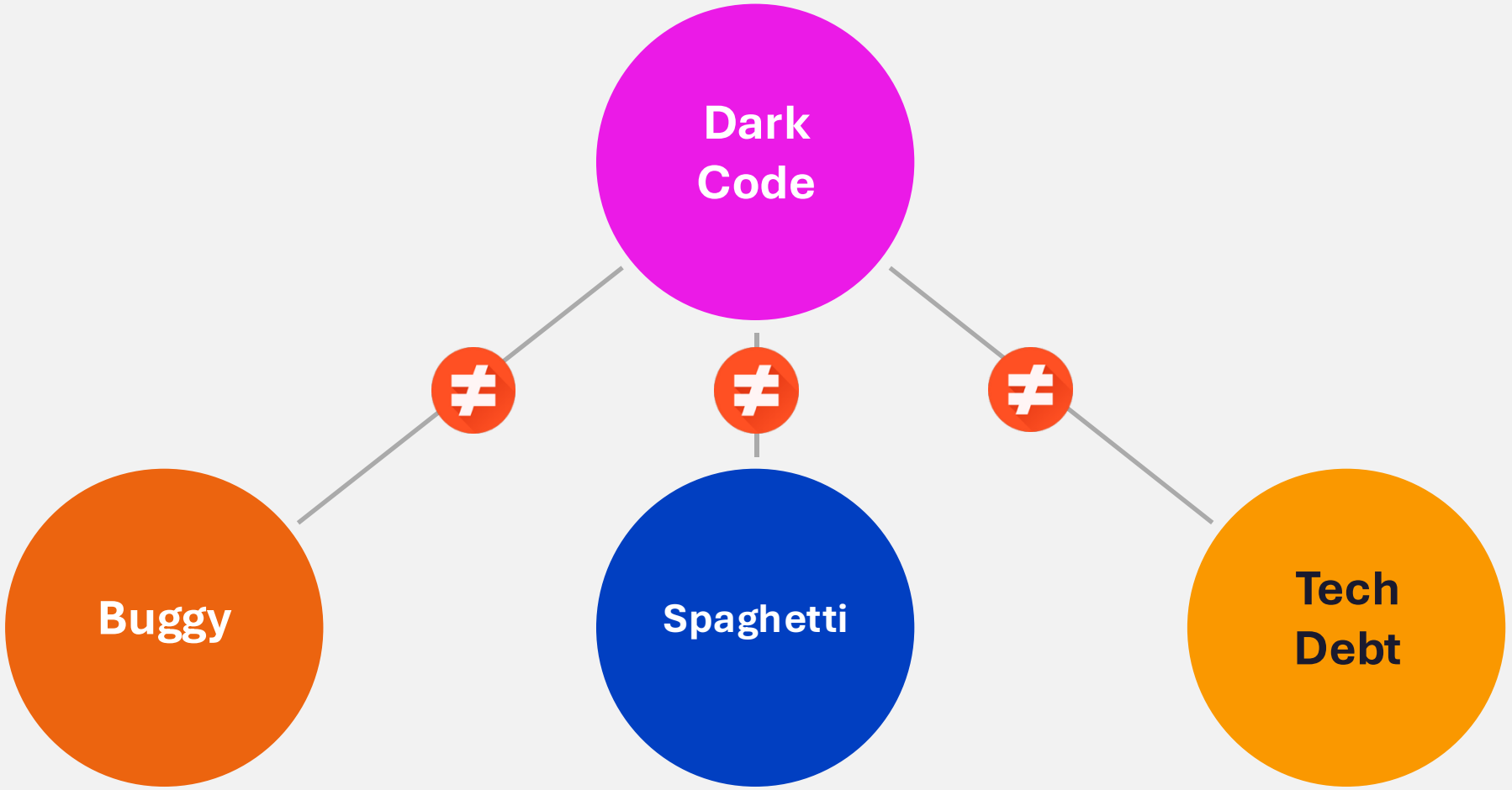
- Complex business logic and workflows
- Integrations with real systems
- Apps that scale and perform
- Production-ready features, fast

What's often forgotten

- Review the output
- Own the codebase
- Understand what was built
- Accountability for what the AI has written

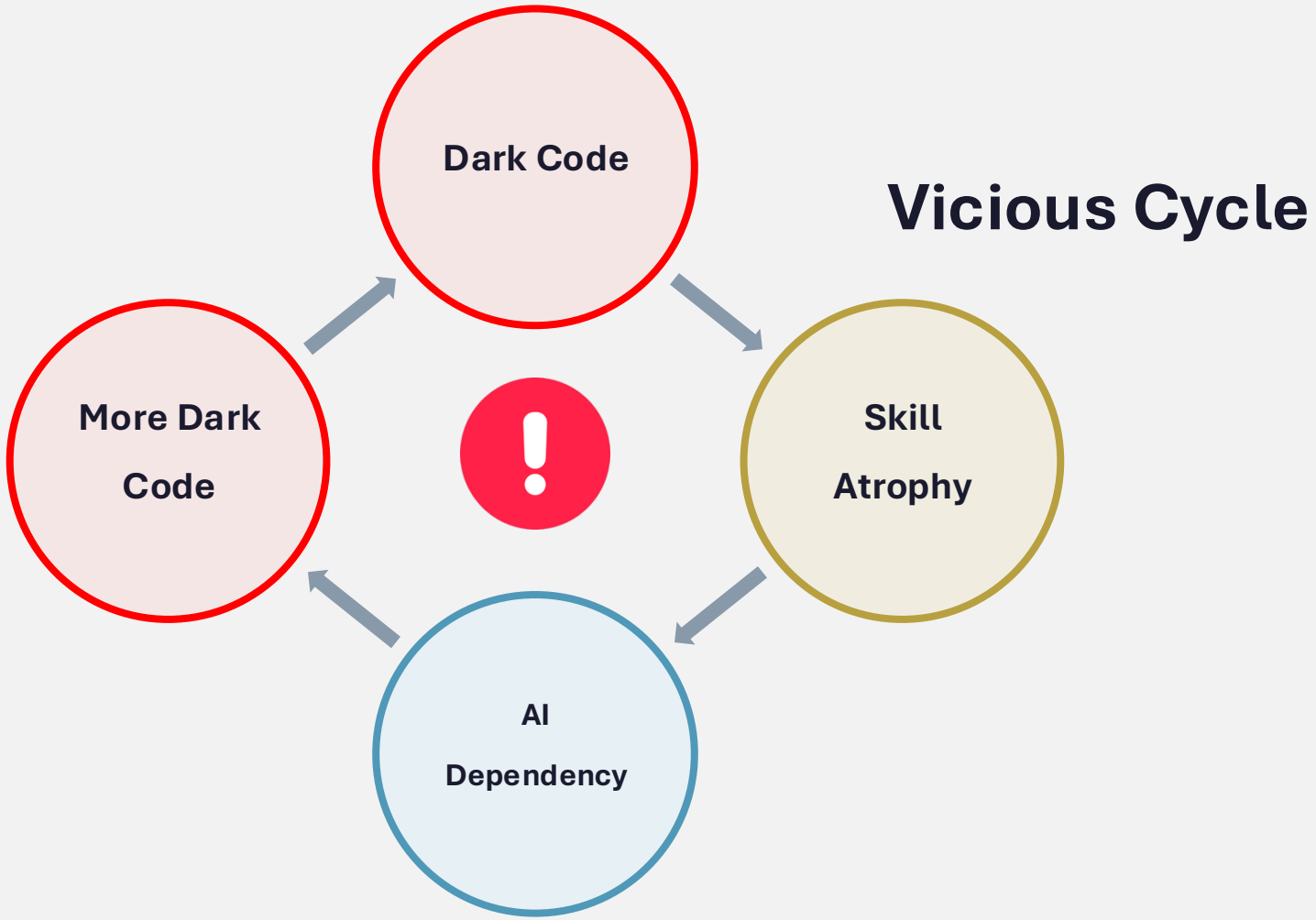
Dark Code

What is Dark Code?



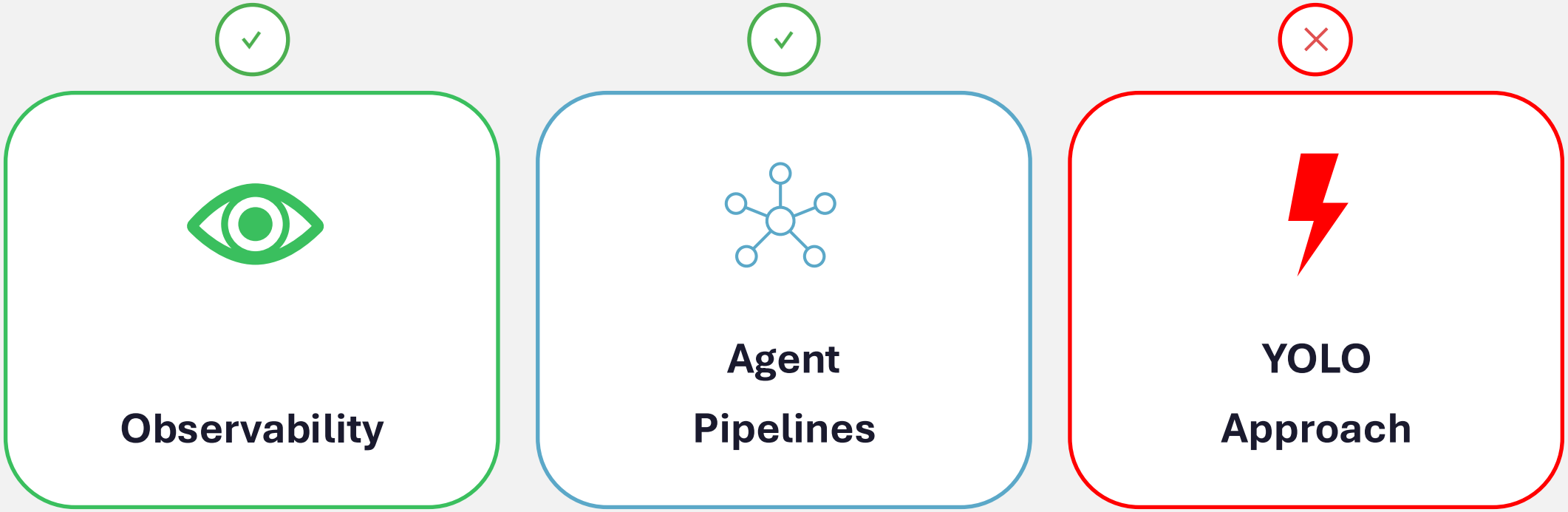
Its code that was never understood by anyone

The Dark Code Trap



More AI dependency creates more *dark code*

Strengths and Weaknesses



Strengths can *mask weaknesses*

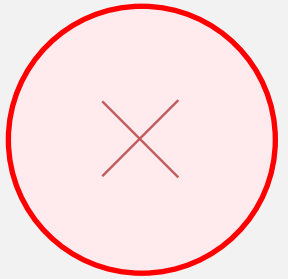
The risk isn’t only technical

Not just technical - organisational and regulatory too

Review is the difference

WITH REVIEW

WITHOUT



You can outsource your thinking, but you can’t outsource your *understanding*

03 Reaching Orbit

How agentic coding works

Your crew, on call

Purpose-built agents that cover the whole delivery lifecycle - each invoked for the job it's best at.

Interviewer

Pulls the real problem out of the user.

Requirements Engineer

Turns intent into concrete specs.

Solution Planner

Maps the approach before code is written.

Developer

Implements the plan, one change at a time.

Diagnostic Fixer

Triages bugs, finds the root cause.

Solution Experts

Security · Performance · Best practices

Test Engineer

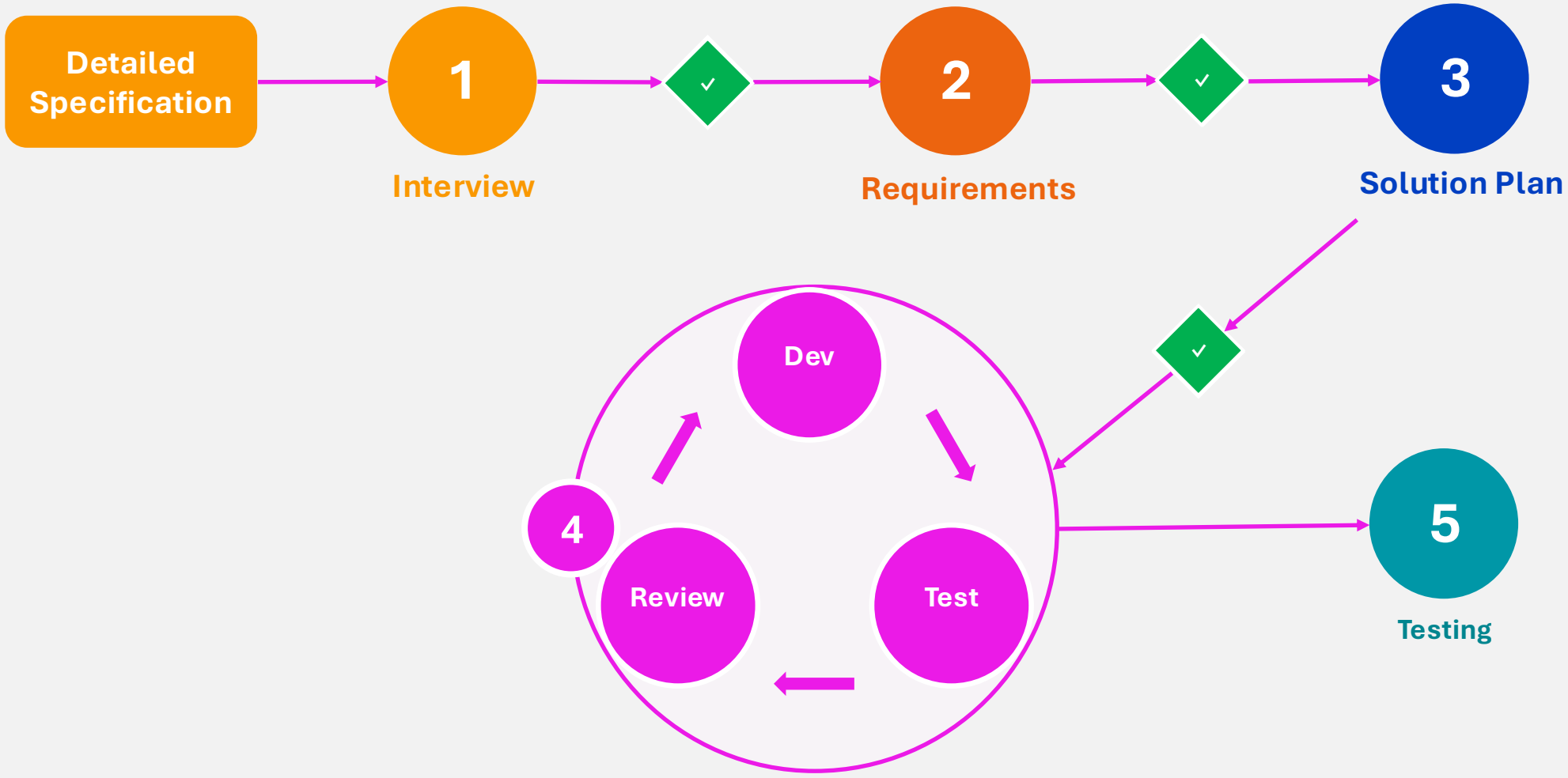
Designs the test strategy - what to prove.

Tester

Runs the suite, reports what happened.

Eight specialists, each doing one thing well.

Spec-driven development lifecycle



Dev → Test → Review

loop until the review gate is passed

The orchestrator holds the plan

Orchestrator

The full picture

- **Project** scope, constraints
- **Plan** architecture, tasks
- **History** decisions, trade-offs
- **State** done, in-flight, next

scoped brief

fresh context

Developer agent

Just the slice it needs

- **Task** one clear job
- **Files** relevant code only
- **Rules** patterns, test reqs

No wider context. No distractions.

Smaller context. Sharper focus. *Fewer mistakes.*

04 Touchdown

Now equip the crew for the terrain ahead

What you can plug in

PLUGIN

Agents

Delegate the whole job.

MCP Tools

Reach beyond the model.

Commands

Act on the surface.

Skills

Playbooks, one slash away.

A ***reusable*** bundle of tools, commands, skills and agents that can be loaded on demand

AL MCP Server

Tools to make your agent, fluent in your codebase.

BUILD & VALIDATE

al_compile
Fast validate, no .app

al_build
Produce .app package

al_publish
Deploy to BC env

UNDERSTAND CODE

al_symbolsearch
Find objects & codeunits

al_downloadsymbols
Refresh symbols

al_getdiagnostics
Errors & warnings

AUTH

al_auth_login
MSAL sign-in for cloud

al_auth_logout
Clear cached tokens

Troubleshooting MCP Server for AL

Runtime-aware help, only when paused.

Trace the path

Full call stack across your code and the framework.

call stack

Inspect live state

Locals, globals and complex types - across any frame.

variables

Read & break

Source for the active frame. Breakpoints by object + line.

source · breakpoints

One developer, multiple agents, in parallel

git worktree turns your repo into a fleet, not a queue.

WITHOUT WORKTREE

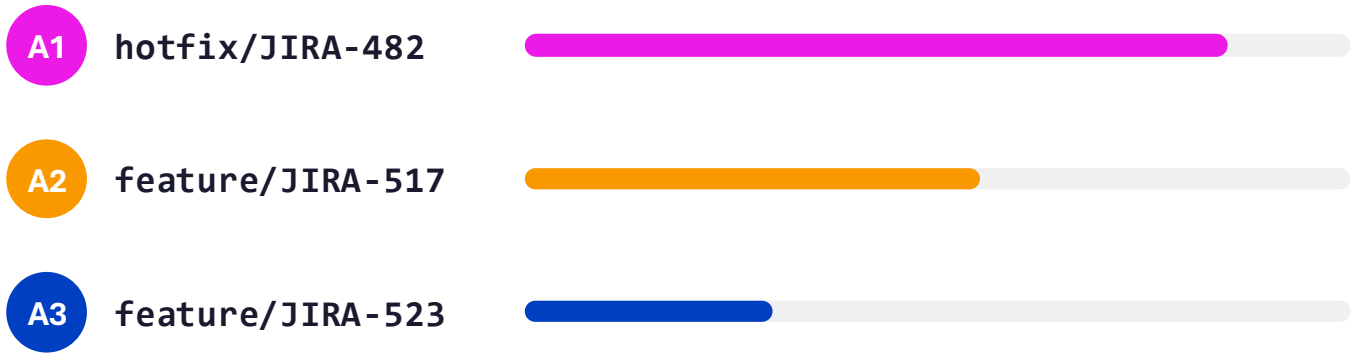
One branch at a time

- **hotfix/JIRA-482**
- feature/JIRA-517
- feature/JIRA-523
- ...queued

stash. checkout. wait. repeat.

WITH GIT WORKTREE

Three branches at once, three agents working



Each in its own directory, its own checkout - no stashing, no context-switching.

The shift is already here

The tools caught up with the ambition. Now the question is what you build with them.

Demo Time

For you
Recent
Starred
Apps
Plans
Spaces
Recent
Digital Pre Sales
Starred
Innovation
Recent
DIG board
OP30870 - SOHOHOUSERETAIL - ...
Radley BAU
OP10012 - HALL WOODHOUSE - U...
OP38255 - HEINEKEN - BC Build
More spaces
Filters
Dashboards
Teams
Customize sidebar

3PL Integration – Publish Standard APIs for Shipping & Receiving with Staging Table Processing

Description

Description:

Design and implement a 3PL (Third-Party Logistics) integration for Business Central that exposes standard APIs for shipping and receiving stock movements. Inbound API payloads should land in staging tables and be processed asynchronously via the Job Queue. Processing must update quantities to ship/receive on the source document, set the relevant posting dates, and post the document.

Background

The 3PL provider requires the ability to confirm shipments and receipts against Business Central documents. Rather than writing directly to BC posting tables, all inbound data must pass through a staging layer to support validation, error handling, and reprocessing without risk to core transactional data.

Scope

API Endpoints

- Publish a standard BC API page for **Shipment Confirmations** (e.g. Sales Shipments, Transfer Shipments, Purchase Return Shipments)
- Publish a standard BC API page for **Receipt Confirmations** (e.g. Purchase Receipts, Transfer Receipts, Sales Return Receipts)
- APIs must follow BC standard API conventions (OData v4, API versioning, appropriate entity naming)
- Support create operations; consider read for status/acknowledgement

Staging Tables

- Create staging table(s) for inbound shipment and receipt records (header + lines)
- Include fields for: Document Type, Document No., Line No., Item No., Quantity, Lot/Serial No. (if applicable), External Reference, 3PL Warehouse Code, Shipment/Receipt Date, Status (Pending / Processing / Processed / Error), Error Message, Created DateTime, Processed DateTime
- Staging records should be immutable once processed; new corrections arrive as new records

To Do | Improve Task

Your pinned fields

Team	None

Details

Assignee	Scott McDonald
Reporter	Scott McDonald
Labels	None
Due date	None
Start date	None
Priority	Medium

Development

More fields

Story Points, Original estimate, Time tracking, Components, Fi...

Automation

Rule executions

Tempo

Open Tempo

Created May 1, 2026 at 3:53 PM
Updated May 1, 2026 at 3:53 PM

04 Avoiding the space debris

Speed without control is how missions end early

It's a tool, not magic

The developer is still the key link in the chain.

The Agent

- Generates & refactors code
- Executes instructions literally
- No business judgement
- Confidently wrong

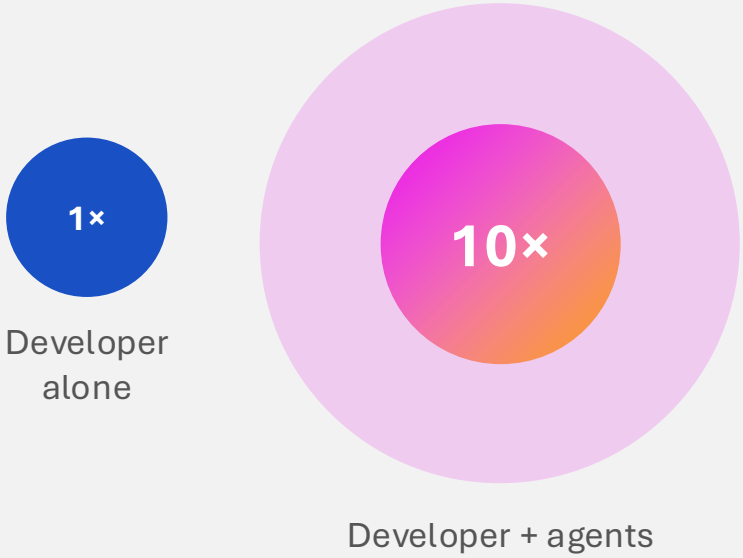
The Developer

- Owns architecture & outcome
- Knows the domain
- Judges “good enough”
- Accountable for what ships

Force multiplier, not replacement. The chain is only as strong as its key link – *you*.

The bigger the multiplier, the bigger the blast radius

More output, more surface area, more places a mistake can hide.



Safeguards

1. **Plan review**
   Before it writes a line - check the approach, the scope, the files it'll touch.
2. **Code review at each gate**
   Every change, read line by line. No rubber stamps, no "looks fine".
3. **Pre-deploy gate**
   Tests green, smoke test done, branch reviewed - then, and only then, UAT.

The role is shifting

Not shrinking

FROM		TO
Writing every line	→	**Directing agents and reviewing output**
Investigating errors manually	→	**Verifying agent diagnoses**
One task at a time	→	**Overseeing parallel workstreams**
Writing tests after the fact	→	**Setting acceptance criteria up front**

Thank You.

Questions